Lecture 4 - Application Framework

"Before software can be reusable it first has to be usable."
- Ralph Johnson

- Event systems
- Engine-level communication between subsystems
- Data structures

Last Week Recap

Game Engine Architecture and subsystems

Event Systems

(1 of) most important concepts ever used in games.

- Manage communication between large modules of a system. Applies to any software.
- Significantly reduces coupling between modules.
- Scalability.

Event Systems (3 fundamental concepts)

- Event
- Listener / handler / subscriber
- Manager / dispatcher / bus

Event Systems (Event)

`Something` of interest to `someone`.

Event Systems (Handler)

Interested parties register with the manager / dispatcher, then listen for events.

Event Systems (Dispatcher)

The (potentially single) party that fires events and notifies interested parties.

Event Systems (Whiteboard example)

Communicating with a module that you don't know anything about.
Possibly a non-existent module even.

Events (High-level Design)

There are three concepts. How are they related? Can we create a schematic diagram of the relationship between them?

Activity (Game engine example)

Using the FXGL codebase:

1. identify the module (directory on GitHub) that deals with **events** (<- big hint).
2. identify the class(es) / method(s) that relate to the three concepts (event, handler, bus).

Usage (Whiteboard example)

Let's consider how to use these three concepts more concretely.
Example use cases from the audience.

Event Systems (Usage, based on design)

```
dispatcher.onEvent(EventType.PLAYER_DIED, {
    showGameOverScreen()
})

var event = Event(EventType.PLAYER_DIED)
dispatcher.fireEvent(event)
```

Activity (Game engine example)

Suppose we have a physics engine and an audio engine. Collisions occur in the physics engine, whereas the sound effects are in the audio engine.
Using an example earlier, construct a solution to the above problem.

Timers

A timer is essentially an interval-based event dispatcher with infinite events.

Timers typically drive the main loop.

Timers

An example timer that runs at ∼60 fps.

```
timer.addAction(mainLoop, Duration.millis(16));
```

Delayed Events

Sometimes, we want to handle events at a later time. Consider an in-game explosive with a timer.

Delayed Events (Example)

```
timer.addDelayedAction(fireEvent, Duration.seconds(3));
```

Targeted Events

Some events may want to target specific game objects. For example, `Open Door` event.

Cancellable Events

Sometimes, the event source (or target) may not exist when the time is up. For example, an enemy that gets killed before their weapon is charged.

Activity

Using the assignment codebase:

1. identify the timer (or alternative) that drives the main loop.
2. identify how events are being fired / handled.

Event Serialization

Events should be easily serializable. This allows easy save data generation, including game object behaviour.

Impact on Gameplay

Consider, a potion effect that speeds up the character. The `onStartEffect` event allows setting the speed to a higher value, then the `onEndEffect` allows setting the value to normal.

Impact on Immersion

Events allow handling of specific scenarios for immersive gameplay. For example, a follower NPC who tells a joke based on the weapon the player has just equipped. The event would be: `onWeaponEquipped(weapon, character)`.

Activity

Find an event type of interest at Skyrim Creation Kit events.
Explain how the event works to a person next to you.

Conclusion

- Event systems form a basis of many application frameworks.

Tutorial

On StudentCentral